NASA Technical Memorandum 110361

Two-Dimensional Mesh Embedding for Galerkin B-Spline Methods

Karim Shariff and Robert D. Moser

August 1995

National Aeronautics and
Space Administration

NASA Technical Memorandum 110361

Two-Dimensional Mesh Embedding for Galerkin B-Spline Methods

Karim Shariff and Robert D. Moser, Ames Research Center, Moffett Field, California

August 1995

National Aeronautics and
Space Administration

Ames Research Center
Moffett Field, California 94035-1000

1 Introduction

The Galerkin B-spline method has a number of useful properties (Moser *et al.* [3]): (i) Thanks to the high degree of continuity of the B-splines, the scheme has high resolution similar to compact finite-differences (Lele [2]) without the need to separately construct numerical boundary schemes. (ii) Constraints such as incompressibility, and coordinate as well as flow singularities can be satisfied *a priori*. Local support allows local constraints as well as boundary conditions to be implemented with ease. (iii) Discretized quantities such as momentum as well as quadratic invariants such as energy (in the inviscid limit) are conserved. (iv) Non-linear terms can be computed without aliasing error. (v) The desired order of accuracy is a parameter.

This work develops another useful property, namely, the ability to do mesh embedding. When a gradient becomes large in a certain direction, structured meshes allow one to cluster grid lines. However, this is efficient only if the region of large gradient spans the domain. This is true only for very simple flows such as laminar shear or shock layers. In more complicated flows considerable savings can be obtained by using the semi-structured approach of embedded meshes of the type depicted in Figure 2. This paper shows how this capability is achieved with B-splines. After providing a brief background on one-dimensional B-splines, it presents an algorithm for choosing an appropriate set of two-dimensional functions and then presents test cases for linear scalar operators. Refinement ratios between mesh blocks are arbitrary and the properties listed above are retained, in particular, conservation with arbitrary order of accuracy. [1] This order of accuracy is uniform everywhere. In contrast, for finite-difference or finite volume methods "simultaneous achievement of both conservation and accuracy is very difficult and even impossible in most cases" according to Kallinderis (1992). In many such methods the order of accuracy drops at mesh interfaces. Such schemes often update each zone separately and interpolate zone boundary information in a separate step. This requires down-wind differencing at some interfaces which can be destabilizing. Such a procedure is not needed here.

Since the cost of computing various matrices is incurred every time the mesh changes, the present method is not recommended for applications requiring frequent adaptive re-meshing. The application we have in mind, namely statistically stationary turbulence, should not require frequent re-meshing. The extension to three-dimensional embedding should be possible in principle, ignoring the cost of mass-matrix inversion.

2 Background on one-dimensional B-splines

For the purposes of this work, a one-dimensional spline is defined to be a polynomial of degree d in each interval with $d-1$ continuous derivatives across interval boundaries. The boundaries of the intervals are called knot-points: the dth derivative of the spline has a jump at the knot points interior to the domain.

A *B-spline* is simply defined as a spline which has support over a minimum number

[1]For uniform knots, resolution (property (i)) is tested using operators whose eigenfunctions oscillate uniformly everywhere. This is not appropriate for embedded meshes and better tests, perhaps involving non-constant coefficients, are needed.

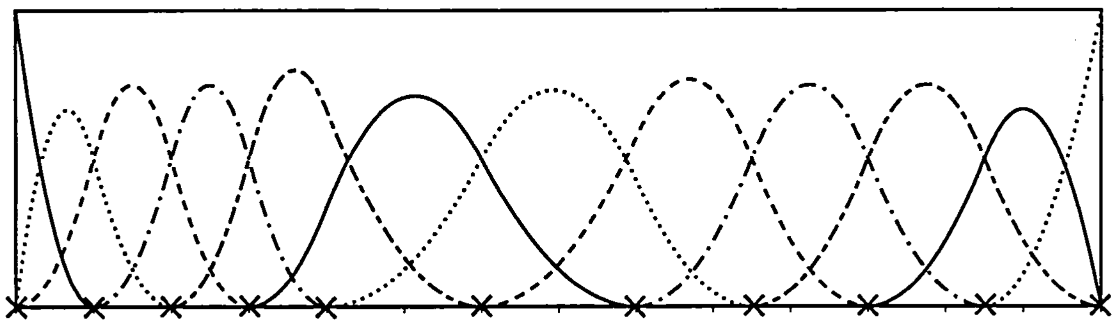

Figure 1: Plot of one-dimensional quadratic B-splines on the set of knot points indicated by $\times$. All functions, except those near the boundary, have support over three intervals. Note: Line types are re-used for different functions.

of intervals and which is normalized. By equating the number of known continuity and normalization conditions to the number of unknown coefficients one finds that the number of minimum intervals is $d + 1$. This is enough information to determine the B-splines for all consecutive $d + 2$ points. Thus a quadratic B-spline has support over three intervals. Figure 1 shows the set of quadratic B-splines for the knot points indicated by $\times$. Near the boundary, the number of continuity conditions available drops and so does the number of intervals of support. To calculate the B-splines ones does not have to actually solve any continuity conditions. Rather, one uses a recurrence relation given in de Boor [1] (Chapter 10) to build up the functions from piecewise constant functions.

Above, the B-splines were merely defined as splines with minimum support. What makes them useful for solving partial differential equations is the result, due to Curry and Schoenberg (see de Boor [1]), that they form a *basis* for spline functions with the given knots.

3 Function selection algorithm

3.1 Mesh Definition

It is assumed that the computational domain is mapped to a rectangle in ξ, η. The case of a more general polygon with right angles (such as a backward-facing step) can perhaps be treated along very similar lines but this is not presently allowed. For convenience, and without loss of generality, the user is required to specify the mesh in terms of sets of points which are swept across certain intervals. For instance in Figure 2, the horizontal lines of the mesh can be generated by sweeping the three sets of η points indicated by • across the ξ intervals indicated. Similarly, the vertical lines can be generated by vertically sweeping the two sets of ξ points indicated by $\times$. Just as in the 1D case where the interval boundaries are knot points, we want mesh lines in the 2D case to represent knot lines, i.e., the lines normal to which the selected functions (of degree d, say) have a jump in the dth normal derivative. Hence we regard each set of points used to sweep out the mesh as being a knot-set with an associated set of one-dimensional functions. The first ξ knot set in Figure 2 is the same as

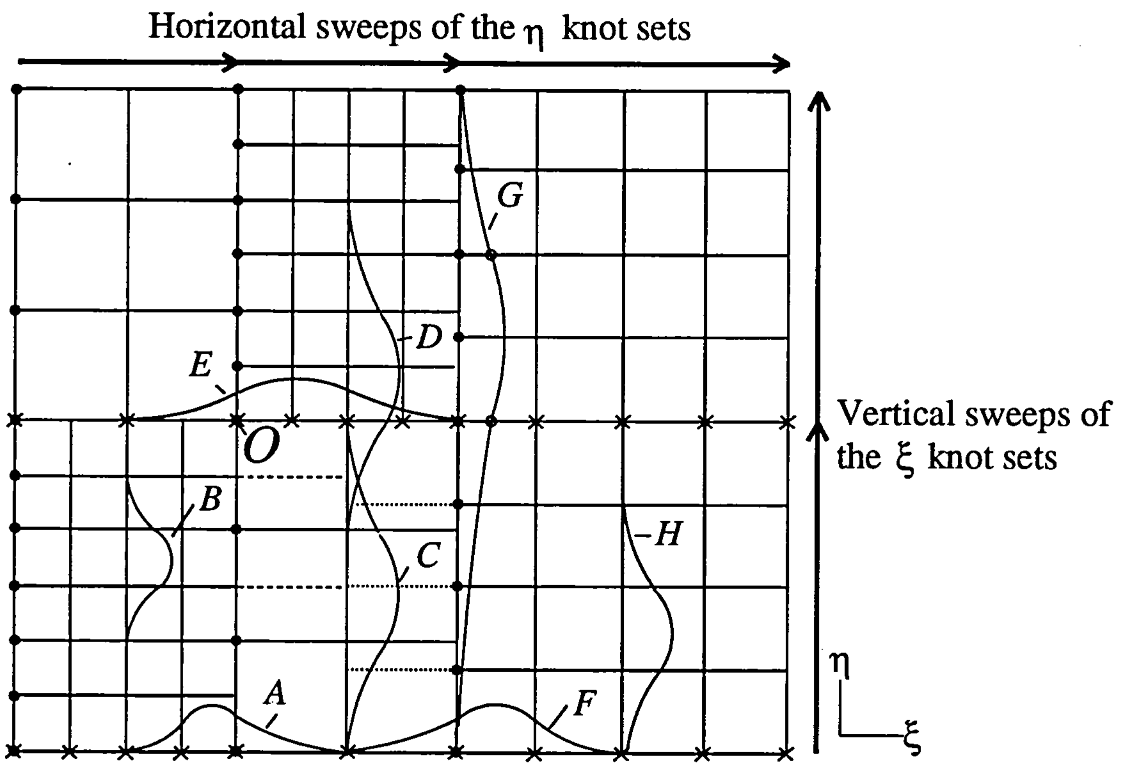
Horizontal sweeps of the η knot sets
Vertical sweeps of the ξ knot sets
G
D
E
O
B
C
H
A
F
η
ξ

Figure 2: Sketch to illustrate the function selection algorithm. All functions are quadratic B-splines and span three knot intervals.

the knot set shown in Figure 1 and has the same associated functions.

If every ξ knot set either contains or is contained by every other ξ knot set (and similarly for η), we will say that the mesh is hierarchical. Figure 2 shows a non-hierarchical mesh and the same algorithm applies for both types of meshes.

It is assumed that the mesh thus specified by sweeping points produces closed cells. Suppose that a vertically swept point ξ_1 "hangs" at η_1, i.e. fails to continue into the next sweep. Then to ensure a closed cell η_1 should be a knot point in the η knot set which sweeps over ξ_1 or begins or ends a sweep there.

As a preliminary, break up the mesh into a set of blocks on each of which the mesh is regular. Figure 2 has five blocks. This decomposition is not always unique and should be performed to minimize block boundaries. In the present implementation a provisional set of blocks is first created on the basis of the sweep intervals. These blocks are then merged with neighbors to create larger blocks.

Below, an algorithm with variants is presented for selecting functions. The first one, the intersection procedure, does not allow functions that create knot lines not defined by the mesh. For non-integer refinement ratios, this restriction would produce coarse functions along block boundaries and the less constrained procedures described in §3.3 should be used.

3.2 Intersection procedure

We want to choose a set of functions $B_n(\xi,\eta), n = 1, 2, \ldots N$ that can represent piecewise polynomials of degree d having knot lines that coincide with the mesh. We have no proof that the procedure provides a complete basis. Each $B_n(\xi,\eta)$ is constructed as a product of one-dimensional B-splines, $f(\xi)g(\eta)$, say. One may consider implementing a brute-force algorithm in which all possible pairs of ξ and η functions defined by the swept knot sets are tried and those that create undesired knot lines are rejected. This procedure would not only be too costly but does not allow the choice of functions not defined by the given knot sets which, we shall see, come into play for non-hierarchical meshes.

Two-dimensional functions confined to each block are chosen *a priori* and it remains to select functions that penetrate multiple blocks. They are referred to as "spilling functions". The procedure is to consider every function, say $f(\xi)$, on each sweep of the ξ knot sets and find suitable functions of η as multipliers. The same procedure is repeated for every η function on each sweep of the η knot sets. Constant reference is made to Figure 2.

(1) First, one obtains the η knot set from which multipliers for $f(\xi)$ will be chosen. The ξ-support of $f(\xi)$ will penetrate a certain range of sweep intervals of the η knot sets. For instance the function A penetrates sweeps 1 and 2 of the η knot sets. To prevent creation of new knot lines, we must choose multipliers from the *intersection* of the η knot sets associated with the range of sweeps. We will call this the *compatible knot set* of $f(\xi)$. For instance in Figure 2, the function B (from the first η knot set) is not a compatible multiplier for A because the resulting product has additional knot lines indicated by the dashes. Function C (from the intersection of η knot sets 1 and 2) is a compatible multiplier. Note that if the knot-sets involved in the intersection operation are hierarchical (i.e. each set either contains or is contained in another) then the intersection operation just chooses the coarsest of the sets. In general, however, the compatible function could turn out to be one that does not belong to any of the knot sets used in the mesh definition.
A reviewer suggested the following for clarification. Note that A is the left-most function in ξ knot-set number 1 that will multiply coarse functions in η. Thus functions (such as $A \times C$) that have the y knot spacing of the coarse block penetrate two ξ intervals of the fine block. In general the number of intervals of penetration is the polynomial degree, d. This is analogous to the fact that in one-dimension, functions with non-zero value and derivative at a boundary penetrate d intervals into the interior of the domain.

(2) Only those functions in the compatible knot set that have support in the sweep interval of the knot set of $f(\xi)$ are relevant; we refer to these as *compatible functions.*

(3) Next, further limit the compatible functions in order to prevent block-confined functions, which have been chosen *a priori*, from being selected. The support of $f(\xi) \times$ its η sweep interval is either confined to a block or it is not.
   (a) If it is confined to a block then only those compatible functions that cross the block boundaries in η are allowed.
   (b) If it is not confined to a block (as is the case for function A), then all compatible functions are allowed.

(4) In order to prevent the selection of 2D functions with additional knot lines, it is necessary that the compatibility be *mutual.* For instance D is a compatible function of A, but A

is not a compatible functions of D. This is because function D penetrates sweeps 1 and 2 of the ξ knot sets but the knots of function A do not belong to their intersection. In particular, the following test is applied: do the knots of $f(\xi)$ belong to the compatible knot-set of $g(\eta)$?
The mutual compatibility test needs to be relaxed at non-hierarchical corners. Consider, for instance, the region near O where two fine regions meet at a corner. The only functions that have support at the corner O that do not result in additional knot lines are functions such as $E \times D$, neither of which belongs to any knot-set used in the definition of the mesh. Since only functions on the knot sets used in the mesh definition seek suitable multipliers, such a product would never be chosen. This is overcome by either of two modifications, denoted as M1 and M2. If the mutual compatibility test fails for a prospective 2D function containing a block corner *and* the $g(\eta)$ does not belong to any of the η knot sets penetrated by the support of $f(\xi)$ then: (M1) $g(\eta)$ is chosen, resulting in the creation of new knot lines, or, (M2) The product of $g(\eta)$ and all the functions on the compatible knot set of $g(\eta)$ that have support at the corner are chosen. In this case no new knot lines are created in the context of the intersection procedure. The unmodified procedure is denoted as M0.

(5) Finally, a check is made that a 2D function selected is unique.

3.3 Less constrained procedures

It may be desirable to change resolution gradually, in a non-integer fashion as shown in Figure 2 between the left and right halves of the mesh. If compatible functions for F were chosen according to the intersection procedure, the very coarse function G would result (its knots are indicated by o). To avoid this, one can dispense with the requirement that no new knot lines be created. One simple way of doing this is to use the "densest" instead of the intersection operation to determine the compatible knot set (i.e. choose the set with the most knots). Another is to use the "densest by sweep (DS)" operation, i.e., assemble the compatible knot set by taking the densest set within each sweep. In this case a function such as H would be a valid multiplier for F and the product would create the additional knot lines indicated by dots. The function H is also mutually compatible with F because the knots of F belong to the (single) ξ knot set penetrated by H. The mutual compatibility test was needed in the intersection algorithm to prevent additional knot lines. Here it is necessary to prevent selection of multiple types of products in the same region. For instance, function C would seek multipliers at some point. One sees that F would not be a mutually compatible function because the knots of C do not belong to the DS of the vertical knot sets on which F has support. This is rightly so because $F \times H$ is the type of product we have selected for this region.

In the tests that follow the algorithm used will be denoted by a prefix: I (intersection), D (densest), or DS (densest on a sweep by sweep basis). This will be followed by a suffix (M0, M1 or M2) to denote the modification to the mutual compatibility test.

Algorithm summary: Perhaps the following summary will aid the reader in holding the algorithm firmly in mind. For every ξ function on the knot sets whose ξ support $\times$ η sweep interval is confined to a block, pick as multipliers only those mutually compatible η functions which cross the block boundaries in η since the rest produce 2D functions confined to the block and have been chosen *a priori.* For a ξ function whose ξ support $\times$ η sweep interval is not confined to a block, choose as multipliers *all* mutually compatible functions. Repeat the procedure for all η functions defined by the given knot sets. If required, relax the test of mutual compatibility according to M1 or M2.

3.4 Matrix structure of linear operators

In a Galerkin (weighted residual) method, linear operators (such as the Laplacian and advection operator in the present examples) give rise to matrices of the following general form:

$$(\mathcal{L}_1 B_m, \mathcal{L}_2 B_n) \equiv \int \mathcal{L}_1 B_m \, \mathcal{L}_2 B_n \, dxdy, \quad m, n = 1, 2 \ldots N, \tag{1}$$

where $\mathcal{L}_1$ and $\mathcal{L}_2$ represent linear differential operators. Similarly, operators with a quadratic non-linearity give rise to integrals of triple products. All matrices need be computed once. Since differentiation does not alter the support region of a function, all linear operators produce matrices with the same structure that depends on which pairs of functions overlap. The template for this structure is constructed once. Pairs of functions confined to each block overlap in a simple way and produce the structured part of the matrix: it has $(d+1)^2$ diagonals for a symmetric operator and $(2d+1)^2$ diagonals for a non-symmetric operator. Spilling and block-confined functions that overlap produce scattered matrix elements as do two overlapping spilling functions. With the template in hand, matrix elements are computed using Gauss quadrature with enough points to ensure exact integrals.

Matrix equations were solved using the preconditioned congugate gradient method which requires that the left hand side matrix multiply a vector in each iteration. The action of linear operators is also a matrix multiplication hence it is important to make this efficient. The multiplication of each diagonal vectorizes over the row direction while the template for storing the scattered elements ensures that their multiplication can be accomplished in a certain number of vector multiplies. This number is the maximum number of scattered elements in a row. The template is rearranged to avoid the most obvious bank conflict, namely that of accessing the same element of the multiplying or resultant vector within a certain number of CPU clocks.

4 Test cases

First consider scalar advection at 45° to the x-axis:

$$u_{,t} + c u_{,x'} = u_{,t} + c\,(u_{,x} + u_{,y}) = 0 \text{ on the unit square,} \tag{2}$$

where x' is a coordinate at 45° to the x-axis. The initial profile $u(x', 0)$ is a Gaussian pulse of unit height and e^{-1} half-width of .15. The advection speed, c, is set to unity. The boundary condition imposed on $u_{,t}$ at the in-flow (left and bottom) boundaries corresponds to uniform propagation of a Gaussian pulse. Henceforth, we use as weight functions those B-splines, say $B_m(x, y), m = 1, 2, \ldots N_o$, which vanish where boundary conditions are applied and represent the solution as

$$u(x, y, t) = \sum_{n=1}^{N_o} a_n(t) B_n(x, y) + \sum_{n=N_o+1}^{N} a_n(t) B_n(x, y). \tag{3}$$

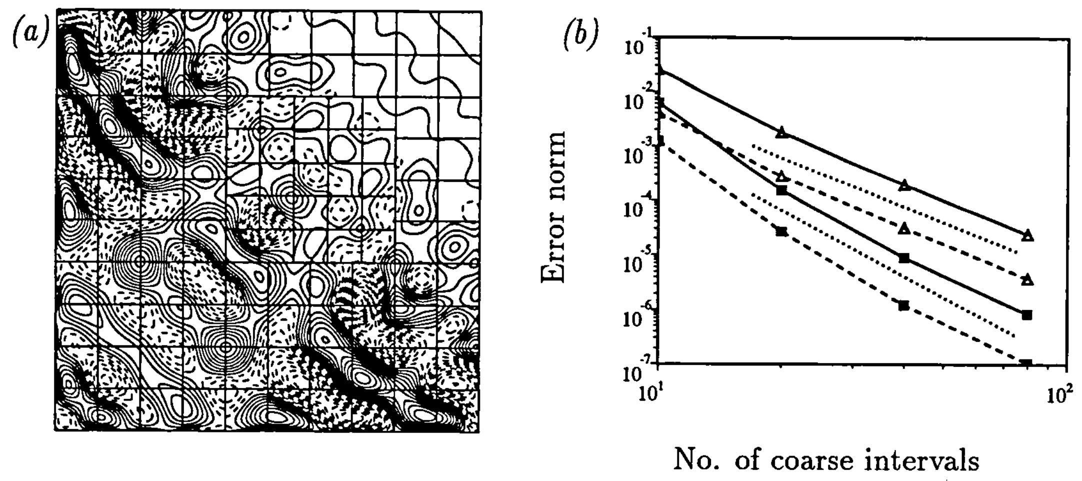

Figure 3: *(a)* Local error for the advection equation after the wave has propagated a distance of 0.6. Contour min., max. and inc. = (−.0065, .0060, .0005). The error is sampled on a 150 × 150 grid. *(b)* Convergence test for the advection equation. Same instant, mesh type and sample points as *(a)*. —— , Maximum error (L_∞ norm); ---- , Average of absolute error (L_1 norm); ········ , reference lines with slope of −3 and −4; △ , for quadratic splines; ▪ , for cubic splines.

Denoting the inner product as (.,.), the weak formulation reads:

$$\sum_{n=1}^{N_o}(B_m, B_n)\dot{a}_n = -c\sum_{n=1}^{N}(B_m, B_{n,x} + B_{n,y})a_n - \sum_{n=N_o+1}^{N}(B_m, B_n)\dot{a}_n, \quad m = 1, 2, \ldots N_o \tag{4}$$

where the coefficients in the last term are known from projecting the boundary specification of $u_{,t}$. In order to make time integration errors smaller than spatial discretization errors, this equation is advanced with a sixth-order Runge-Kutta scheme with cfl $\equiv c\Delta t/\Delta x_{\min} =$ 0.5. Figure 3*a* shows the local error obtained using cubic splines and algorithm DM0 for a non-integer refinement ratio of 5/4. At the instant shown, the pulse has propagated 4 half-widths and the peak of the pulse lies on the diagonal of the fine block. The error is perfectly symmetric about the 45° line, smooth, without any peculiarities at the mesh interface, and attains a maximum of 0.65% even on the rather coarse mesh. Figure 3*b* is the result of a convergence test using quadratic and cubic splines. Aside from a little curvature and flattening for the cubic case, the approximation-theoretic convergence rate (polynomial degree +1) is obtained.

Next consider the Poisson equation for the streamfunction given the vorticity, ω:

$$\nabla^2\psi(x, y) = -\omega(x, y) \text{ on the unit square } \Omega \text{ with } \psi = g(s) \text{ on } \partial\Omega. \tag{5}$$

In order to eliminate unknown boundary terms in the weak formulation and to impose the Dirichlet boundary condition strongly we again use as weight functions all the B-splines

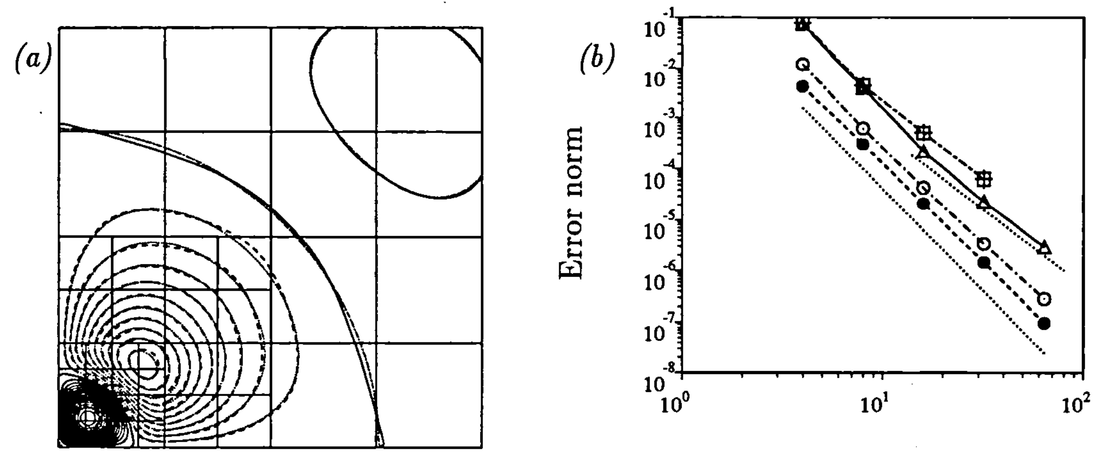

Figure 4: *(a)* Solution to the three-vortex Poisson equation sampled on a 100×100 grid. The actual mesh is shown. Computed solution: ---- , negative contours; —— , positive contours. Exact solution: ········ , negative contours; — — , positive contours.. *(b)* Convergence for the three-vortex Poisson equation test. Error sampled at 512×512 points: — — □ , Maximum error for the embedded mesh; — — + , Maximum error for a uniform mesh having the finest spacing of the embedded mesh. Error sampled at all the "mesh-points": —— △ , maximum error (L_∞ norm); —·— ○ , r.m.s. error (L_2 norm); ---- ● , Average of absolute error (L_1 norm). ········ , reference lines with slopes of -3 and -4.

which vanish at the boundary. The weak formulation reads:

$$\sum_{n=1}^{N_o} c_n(\nabla B_m, \nabla B_n) = (B_m, \omega) - \sum_{n=N_o+1}^{N} c_n(\nabla B_m, \nabla B_n), \quad m = 1, 2, \ldots N_o \qquad (6)$$

The test vorticity field in the present example consists of three axisymmetric Gaussians of alternating sign and graded intensity superposed with images to make the left and bottom boundaries impermeable walls. The exact streamfunction due to each vortex can be expressed in terms of the exponential integral. In the numerical solution the condition $\psi = 0$ is imposed on the left and bottom walls and the exact solution is imposed on the top and right boundaries.

Figure 4*a* shows that even for the very coarse mesh, the agreement between the exact and numerical solutions is excellent (the intersection algorithm (IM0) with quadratic B-splines was used). For comparison, the solution was also obtained for a uniform mesh having everywhere the mesh spacing of the finest block of the embedded mesh. The maximum error occurs in the intense vortex near the corner and it has virtually the same value for the two meshes (compare □ and + in Figure 4*b*; the error was sampled on a 512×512 set of points for both meshes).

The rest of the curves in Figure 4*b* show that when the error is evaluated at the "mesh-points" (i.e. the intersection points of the knot lines), two of the norms converge at fourth order which is one order higher than the approximation theoretic result. This phenomenon is called super-convergence and deserves a brief comment. Thomée [4] showed that in one-dimension and with periodic boundary conditions, the Galerkin B-spline method exhibits super-convergence at the knot points. In particular for the heat equation the convergence rate is $2d$ while for the advection equation it is $2d+2$. Exact time integration is assumed in both cases. Various numerical tests were performed and they indicated that super-convergence was weak for two dimensions with non-periodic boundary conditions: Figure 4*b* represents, indeed, an exceptional case. For the advection equation there is no super-convergence even on a uniform mesh. For the Poisson equation on a uniform mesh, the odd-degree splines considered did not exhibit superconvergence, while among the even-degree splines only the quadratic functions displayed a rate consistent with Thomée's result. Quartics converged at sixth order compared with fifth order for approximation theory and the eighth order of Thomée's result.

In the advection test the local error had no peculiarities near the mesh interface. The same is not true for the Poisson test. Figure 5*a* shows that the maximum error occurs in the region of the intense vortex nearest the corner, as one would expect, but increases again in the form of positive and negative layers along the interface with diminishing amplitude normal to the interface. A regular mesh (without embedding; see Figure 5*b*) but with the same change in spacing *normal* to the interface as the embedded mesh gives similar features in the error but the peak is a little smaller (70% of the value in the embedded mesh). In Figure 5*c* normal spacing is kept uniform while *tangential* resolution changes. The pattern of the error near the interface resembles swords pointing normal to the interface with the positive peak being half the value in the original embedded mesh. Figure 5*d* shows that near the interface of interest, the two errors very nearly add to give the error in the original embedded mesh.

The final test is one of robustness: we consider a problem for which a uniform mesh is optimal and study how much the resolution degrades when the mesh is dislocated (as shown in Figure 6*b*) and the embedding procedure is applied. A uniform mesh is optimal for problems in which the solution oscillates uniformly everywhere in the domain. The eigenvalue problem for the Laplacian operator is one such problem:

$$\nabla^2\psi = \lambda\psi \text{ on a rectangle of unit width and height } h, \text{ with } \frac{\partial\psi}{\partial n} = 0 \text{ on the boundary.} \quad (7)$$

The von Neumann boundary condition was chosen because it makes the boundary term of the weak formulation vanish. The square, h^2, of the height of the domain is chosen to be irrational ($h^2 = \sqrt{2}$) to avoid degenerate eigenvalues. The exact eigensolutions are:

$$\lambda_{m,n} = -\pi^2(m^2 + (n/h)^2), \quad \psi_{m,n} = \cos(m\pi x)\cos(n\pi y), \quad (8)$$

The extent to which a numerical method is able reproduce the exact eigenvalues and eigenfunctions is a test of its resolution of the Laplace operator (with the given boundary conditions) across all scales. We compare performance on a uniform 10×10 interval mesh (without embedding) with performance on the dislocated mesh shown in Figure 6*b*.

The pairing of a numerical eigensolution with an exact one is somewhat arbitrary for the inaccurate eigensolutions. The procedure used was to evaluate each numerical eigenfunction

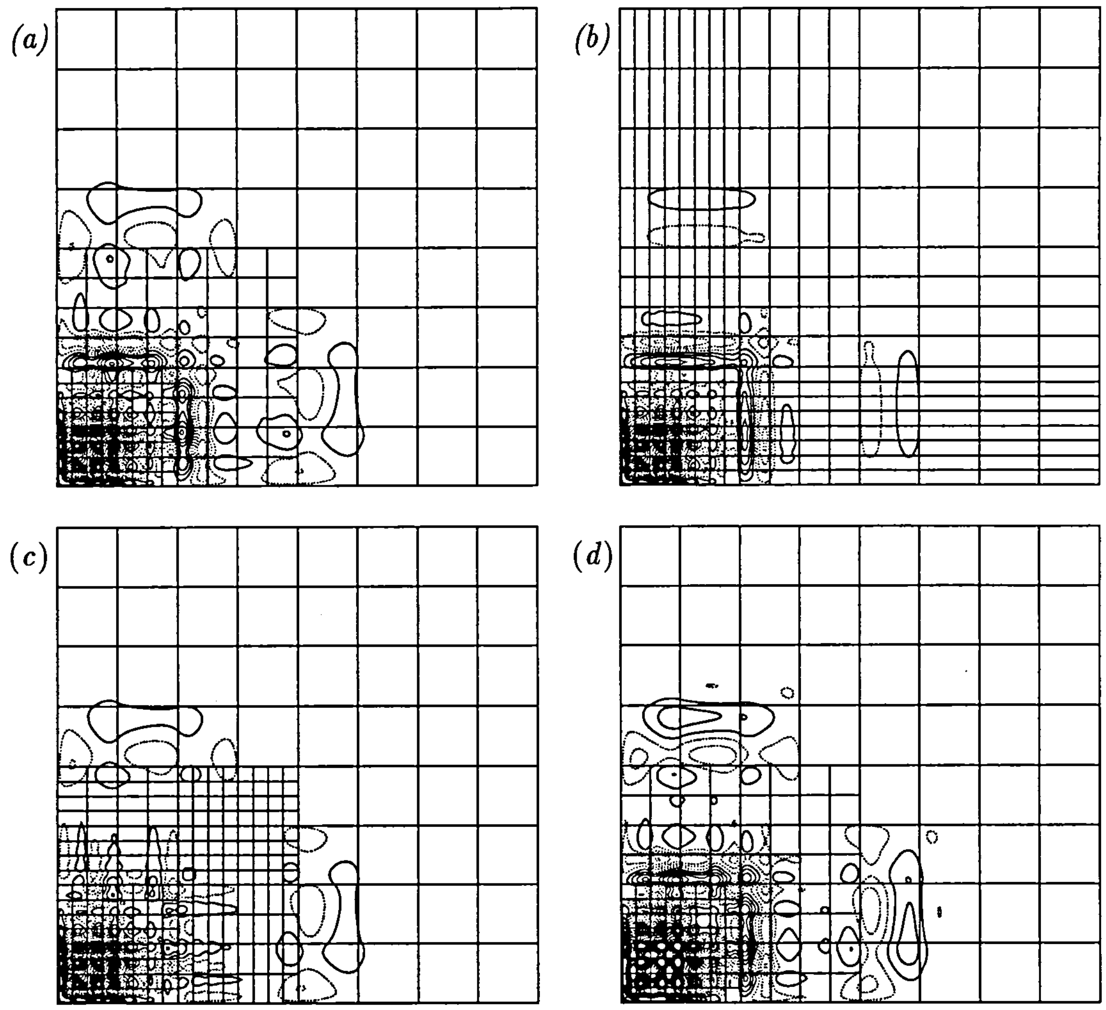

Figure 5: Local error on three different meshes for the three-vortex Poisson equation test. In all plots, contour min., max., inc. = $(-.00475, .00425, .00050)$. This makes the lowest contour levels $\pm.00025$ rather than zero. —— , positive values; ········ , negative values. The error is evaluated on a 300×300 grid. *(a)* Mesh embedding with abrupt changes in spacing normal and tangential to the mesh interfaces. *(b)* A non-embedded but non-uniform mesh with abrupt changes in normal resolution only. *(c)* Mesh embedding with an abrupt change of spacing in the tangential direction in the region of interest. *(d)* The result of adding *(b)* and *(c)*.

Table 1: Number of eigenvalues (n_λ) and eigenfunctions (n_v) with error less than the specified tolerances. Relative error is used for eigenvalues; L_2 error is used for eigenfunctions.

Mesh type	10% error tolerance	1% error tolerance	No. of degrees of freedom
Uniform	$n_\lambda = 90$ $n_v = 63$	$n_\lambda = 30$ $n_v = 22$	144
Dislocated	$n_\lambda = 93$ $n_v = 57$	$n_\lambda = 29$ $n_v = 22$	151

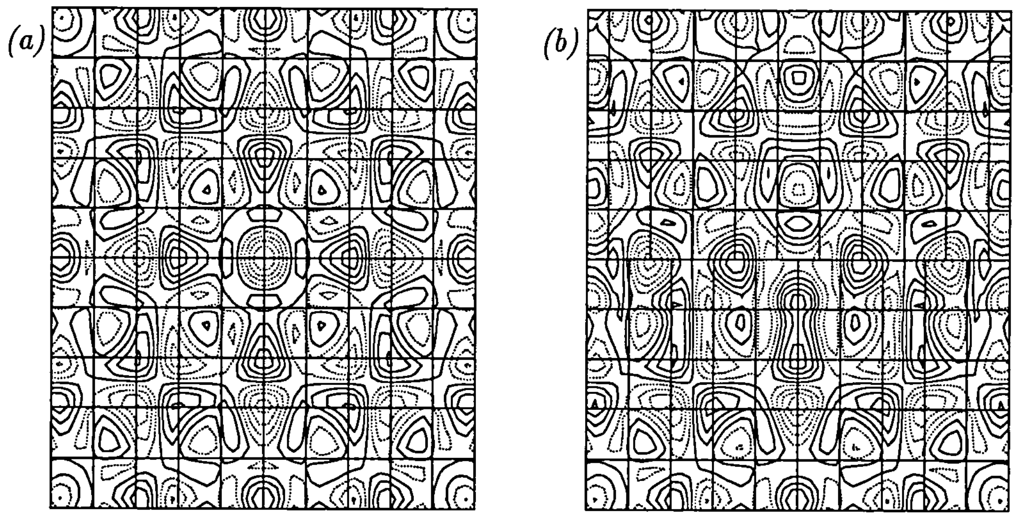

Figure 6: Local error for the eigenfunction for $m = n = 6$. In both plots contour min., max., inc. = $(-.19, .17, .04)$ which makes the lowest contour levels $-.03$ and $.01$ rather than zero. —— , positive values; ········ , negative values. The error is evaluated on a 50×50 grid. *(a)* Uniform mesh *(b)* Dislocated mesh.

on a 50×50 grid, normalize it to have a value of unity at the origin and pair it with the closest (in the discrete L-2 sense) unpaired exact eigensolution. The pairing proceeded in order of increasing numerical eigenvalue. To assess resolution, the number of eigenvalues and eigenvectors which satisfy a given error tolerance for the two meshes are shown in Table 1. For a tolerance of .01, the number of "good" eigensolutions is only slightly lower for the dislocated mesh. The overhead of extra functions at the dislocation increases the number of degrees of freedom from 144 to 151. Therefore the ratio of good eigenvectors to the total number of degrees of freedom degrades by 5% for a tolerance of .01. For a tolerance of .10 this ratio degrades by 14%. Figure 6 compares the error in the eigenfunction which has three wavelengths over the ten intervals in each direction. The dislocated mesh actually has a maximum error which is smaller by 3% but L_2 error which is larger by 3%.

5 Conclusion

A technique has been developed for achieving two-dimensional mesh embedding in the context of a Galerkin method with B-splines as basis functions. The results of test cases are encouraging and work is under way to apply the technique to the incompressible Navier-Stokes equations for three-dimensional flow in two-dimensional curvilinear coordinates.

Acknowledgement

The authors thank Drs. Richard Beam, Dennis Jespersen, Marshal Merriam and Michael Rogers and Lewis Schiff reviewing the manuscript.

References

[1] C. de Boor. *A Practical Guide to Splines.* Springer-Verlag, 1978.

[2] S.K. Lele. Compact finite-difference schemes with spectral-like resolution. *J. Comp. Phys.*, 103:16–42, 1992.

[3] R.D. Moser, K. Shariff, P. Loulou, and A. Kravchenko. B-spline Galerkin methods with applications to the Navier-Stokes equations. In preparation.

[4] V. Thomée. Spline-Galerkin methods for initial-value problems with constant coefficients. In A. Dold and B. Eckmann, editors, *Conference on the Numerical Solution of Differential Equations*, volume 363 of *Lecture Notes in Mathematics.* Springer-Verlag, 1974.

REPORT DOCUMENTATION PAGE	*Form Approved* *OMB No. 0704-0188*

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE August 1995	3. REPORT TYPE AND DATES COVERED Technical Memorandum

4. TITLE AND SUBTITLE Two-Dimensional Mesh Embedding for Galerkin B-Spline Methods	5. FUNDING NUMBERS 505-59-53
6. AUTHOR(S) Karim Shariff and Robert D. Moser	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Ames Research Center Moffett Field, CA 94035-1000	8. PERFORMING ORGANIZATION REPORT NUMBER A-950083
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) National Aeronautics and Space Administration Washington, DC 20546-0001	10. SPONSORING/MONITORING AGENCY REPORT NUMBER NASA TM-110361

11. SUPPLEMENTARY NOTES

Point of Contact: Karim Shariff, Ames Research Center, MS 202A-1, Moffett Field, CA 94035-1000; (415) 604-5361

12a. DISTRIBUTION/AVAILABILITY STATEMENT Unclassified — Unlimited Subject Category 34	12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

A number of advantages result from using B-splines as basis functions in a Galerkin method for solving partial differential equations. Among them are arbitrary order of accuracy and high resolution similar to that of compact schemes but without the aliasing error. This work develops another property, namely, the ability to treat semi-structured embedded or zonal meshes for two-dimensional geometries. This can drastically reduce the number of grid points in many applications. Both integer and non-integer refinement ratios are allowed. The report begins by developing an algorithm for choosing basis functions that yield the desired mesh resolution. These functions are suitable products of one-dimensional B-splines. Finally, test cases for linear scalar equations such as the Poisson and advection equation are presented. The scheme is conservative and has uniformly high order of accuracy throughout the domain.

14. SUBJECT TERMS B-splines, Mesh embedding, Galerkin methods			15. NUMBER OF PAGES 15 16. PRICE CODE A03
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT